THIS PAGE BLANK (USPTO)

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

| | | |
|---|---|---|
| (51) International Patent Classification 6 : H01J 3/26 | A1 | (11) International Publication Number: WO 98/20511<br>(43) International Publication Date: 14 May 1998 (14.05.98) |

(21) International Application Number: PCT/US97/19630

(22) International Filing Date: 24 October 1997 (24.10.97)

(30) Priority Data:
08/742,994 1 November 1996 (01.11.96) US

(71) Applicant: PACKETEER, INC. [US/US]; Suite 305, 307 Orchard City Drive, Campbell, CA 95008 (US).

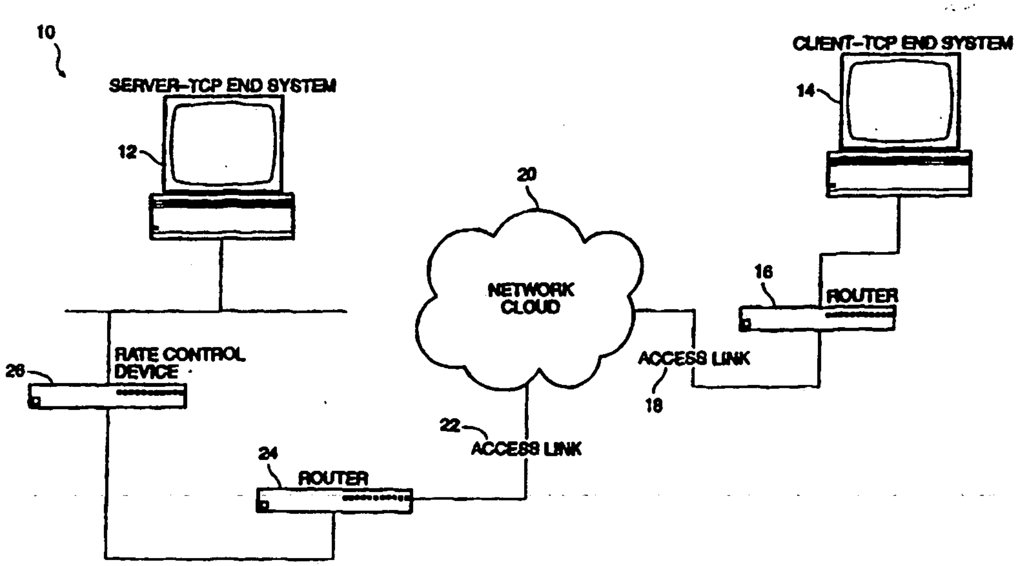
(72) Inventor: PACKER, Robert, L.; 16095 Redwood Lodge Road, Los Gatos, CA 95036 (US).

(74) Agents: ALLEN, Kenneth, R. et al.; Townsend and Townsend and Crew LLP, 8th floor, Two Embarcadero Center, San Francisco, CA 94111-3834 (US).

(81) Designated States: AU, CN, IL, JP, KR, NZ, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

**Published**
*With international search report.*
*Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.*

(54) Title: METHOD FOR EXPLICIT DATA RATE CONTROL IN A PACKET COMMUNICATION ENVIRONMENT WITHOUT DATA RATE SUPERVISION




**(57) Abstract**

A method for explicit data rate control is introduced into a packet communication environment (10) which does not have data rate supervision by adding latency to the acknowledgement (ACK) packet and by adjusting the size of the flow control window associated with the packet in order to directly control the data rate of the source data at the station (12 or 14) originating the packet.

*FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **AL** | Albania | **ES** | Spain | **LS** | Lesotho | **SI** | Slovenia |
| **AM** | Armenia | **FI** | Finland | **LT** | Lithuania | **SK** | Slovakia |
| **AT** | Austria | **FR** | France | **LU** | Luxembourg | **SN** | Senegal |
| **AU** | Australia | **GA** | Gabon | **LV** | Latvia | **SZ** | Swaziland |
| **AZ** | Azerbaijan | **GB** | United Kingdom | **MC** | Monaco | **TD** | Chad |
| **BA** | Bosnia and Herzegovina | **GE** | Georgia | **MD** | Republic of Moldova | **TG** | Togo |
| **BB** | Barbados | **GH** | Ghana | **MG** | Madagascar | **TJ** | Tajikistan |
| **BE** | Belgium | **GN** | Guinea | **MK** | The former Yugoslav Republic of Macedonia | **TM** | Turkmenistan |
| **BF** | Burkina Faso | **GR** | Greece | | | **TR** | Turkey |
| **BG** | Bulgaria | **HU** | Hungary | **ML** | Mali | **TT** | Trinidad and Tobago |
| **BJ** | Benin | **IE** | Ireland | **MN** | Mongolia | **UA** | Ukraine |
| **BR** | Brazil | **IL** | Israel | **MR** | Mauritania | **UG** | Uganda |
| **BY** | Belarus | **IS** | Iceland | **MW** | Malawi | **US** | United States of America |
| **CA** | Canada | **IT** | Italy | **MX** | Mexico | **UZ** | Uzbekistan |
| **CF** | Central African Republic | **JP** | Japan | **NE** | Niger | **VN** | Viet Nam |
| **CG** | Congo | **KE** | Kenya | **NL** | Netherlands | **YU** | Yugoslavia |
| **CH** | Switzerland | **KG** | Kyrgyzstan | **NO** | Norway | **ZW** | Zimbabwe |
| **CI** | Côte d'Ivoire | **KP** | Democratic People's Republic of Korea | **NZ** | New Zealand | | |
| **CM** | Cameroon | | | **PL** | Poland | | |
| **CN** | China | **KR** | Republic of Korea | **PT** | Portugal | | |
| **CU** | Cuba | **KZ** | Kazakstan | **RO** | Romania | | |
| **CZ** | Czech Republic | **LC** | Saint Lucia | **RU** | Russian Federation | | |
| **DE** | Germany | **LI** | Liechtenstein | **SD** | Sudan | | |
| **DK** | Denmark | **LK** | Sri Lanka | **SE** | Sweden | | |
| **EE** | Estonia | **LR** | Liberia | **SG** | Singapore | | |

# METHOD FOR EXPLICIT DATA RATE CONTROL IN A PACKET COMMUNICATION ENVIRONMENT WITHOUT DATA RATE SUPERVISION

## COPYRIGHT NOTICE



## BACKGROUND OF THE INVENTION

This invention relates to digital packet telecommunications, and particularly to data flow rate control at a particular layer of a digitally-switched packet telecommunications environment normally not subject to data flow rate control wherein data packets are communicated at a variety of rates without supervision as to rate of data transfer, such as under the TCP/IP protocol suite.

The widely-used TCP/IP protocol suite, which implements the world-wide data communication network environment called the Internet and is employed in local networks also (Intranets), intentionally omits any explicit supervisory function over the rate of data transport over the various media which comprise the network. While there are certain perceived advantages, this characteristic has the consequence of juxtaposing very high-speed packets and very low-speed packets in potential conflict and the resultant inefficiencies. Certain loading conditions can even cause instabilities which could lead to overloads that could stop data transfer temporarily. It is therefore considered desirable to provide some mechanism to optimize efficiency of data transfer while minimizing the risk of data loss.

In order to understand the exact context of the invention, an explanation of technical aspects of the Internet/Intranet telecommunications environment may prove helpful.

Internet/Intranet technology is based largely on the TCP/IP protocol suite, where IP is the network level Internet Protocol and TCP is the transport level Transmission Control Protocol. At the network level, IP provides a "datagram" delivery service. By contrast, TCP builds a transport level service on top of the datagram service to provide guaranteed, sequential delivery of a byte stream between two IP hosts.

TCP has 'flow control' mechanisms operative at the end stations only to limit the rate at which a TCP endpoint will emit data, but it does not employ explicit data rate control. The basic flow control mechanism is a 'sliding window', a time slot within an allowable window which by its sliding operation essentially limits the amount of unacknowledged transmit data that a transmitter can emit.

Another flow control mechanism is a congestion window, which is a refinement of the sliding window scheme involving a conservative expansion to make use of the full, allowable window. A component of this mechanism is sometimes referred to as 'slow start'.

The sliding window flow control mechanism works in conjunction with the Retransmit Timeout Mechanism (RTO), which is a timeout to prompt a retransmission of unacknowledged data. The timeout length is based on a running average of the Round Trip Time (RTT) for acknowledgment receipt, i.e. if an acknowledgment is not received within (typically) the smoothed RTT + 4*mean deviation, then packet loss is inferred and the data pending acknowledgment is retransmitted.

Data rate flow control mechanisms which are operative end-to-end without explicit data rate control draw a strong inference of congestion from packet loss (inferred, typically, by RTO). TCP end systems, for example, will 'back-off', i.e., inhibit transmission in increasing multiples of the base RTT average as a reaction to consecutive packet loss.

Bandwidth Management in TCP/IP Networks

Bandwidth management in TCP/IP networks is accomplished by a combination of TCP end systems and routers which queue packets and discard packets when some congestion threshold is exceeded. The discarded and therefore unacknowledged packet serves as a feedback mechanism to the TCP transmitter. (TCP end systems are clients or servers running the TCP transport protocol, typically as part of their operating system.)

The term 'bandwidth management' is often used to refer to link level bandwidth management, e.g. multiple line support for Point to Point Protocol (PPP). Link level bandwidth management is essentially the process of keeping track of all traffic and deciding whether an additional dial line or ISDN channel should be opened or an extraneous one closed. The field of this invention is concerned with *network* level bandwidth management, i.e. policies to assign available bandwidth from a single logical link to network flows.

Routers support various queuing options. These options are generally intended to promote fairness and to provide a rough ability to partition and prioritize separate classes of traffic. Configuring these queuing options with any precision or without side effects is in fact very difficult, and in some cases, not possible. Seemingly simple things, such as the length of the queue, have a profound effect on traffic characteristics. Discarding packets as a feedback mechanism to TCP end systems may cause large, uneven delays perceptible to interactive users.

Routers can only control outbound traffic. A 5% load or less on outbound traffic can correspond to a 100% load on inbound traffic, due to the typical imbalance between an outbound stream of acknowledgements and an inbound stream of data.

A mechanism is needed to control traffic which is more efficient in that it is more tolerant of and responsive to traffic loading.

As background, further information about TCP/IP and the state of the art of flow control may be had in the following publications:

-Comer, Douglas. *Internetworking with TCP/IP. Vol I.* Prentice Hall, 1991.

-Comer, Douglas and Stevens, David. *Internetworking with TCP/IP. Vol II. Design, Implementation, and Internals.* Prentice Hall, 1991.

-W. Richard Stevens, *TCP/IP Illustrated. Vol. I - The Protocols.* Addison-Wesley. 1994.

-RFC 793. *Transmission Control Protocol.* Postel, 1981.

-RFC 1122. *Host Requirements.* Braden 1989.

A particularly relevant reference to the present work is:

-Hari Balakrishnan, Srinivasan Seshan, Elan Amir, Randy H. Katz. *Improving TCP/IP Performance over Wireless Networks.* Proc. 1st ACM Conf. on Mobile Computing and Networking, Berkeley, CA, November 1995.

The above document reports efforts of a research group at the University of California at Berkeley to implement TCP 'interior spoofing' to mitigate the effects of single packet loss in micro-cellular wireless networks. Its mechanism is the buffering of data and performing retransmissions to preempt end to end RTO. It is a software mechanism at a wireless network based station which will aggressively retry transmission a single time when it infers that a single packet loss has occurred. This is a more aggressive retransmission than the normal TCP RTO mechanism or the 'quick recovery' mechanism, whereby a transmitter retransmits after receiving N consecutive identical acknowledgments when there is a window of data pending acknowledgment.

Sliding window protocols are known, as in Comer, Vol. I. page 175. Known sliding window protocols are not time based. Rate is a byproduct of the characteristics of the network and the end systems.

## SUMMARY OF THE INVENTION

According to the invention, a method for explicit network level data rate control is introduced into a level of a packet communication environment at which there is a lack of data rate supervision to control assignment of available bandwidth from a single logical link to network flows. The method includes adding latency to the acknowledgement (ACK) packet of the *network level* and adjusting the reported size of the existing flow control window associated with the packet in order to directly control the data rate of the source data at the station originating the packet.

Called direct feedback rate control, the method comprises a mechanism that mitigates TCP packet level traffic through a given link in order to manage the bandwidth of that link. A software mechanism to implement the function may be a software driver, part of a kernel of an operating system or a management function implemented on a separate dedicated machine in the communication path.

The invention has the advantage of being transparent to all other protocol entities in a TCP/IP network environment, For example,in the connections controlled according to the invention, it is transparent to TCP end systems (ie., end systems using the TCP protocol).

The invention will be better understood upon reference to the following detailed description in connection with the accompanying drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a system according to the invention.

## DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

Referring to Fig. 1, a system 10 which uses the invention comprises a first TCP end system 12, such as a server device, a second TCP end system 14, such as a client device, which is connected through a first router 16 and thence via a first access link 18 into a network cloud 20 of a character as hereinafter explained, which in turn provides a logical

connection via a second access link 22 to a second router 24. According to the invention, there is provided between at least one of the end systems 12 and one of the routers 24 a rate control device 26 which is operative to control the rate at which a TCP transmitter can emit packets. In this invention, rate control is accomplished by (1) delaying the transmission of an acknowledgement, which may be a special packet, (known as an ACK) issued by the end system 12 or 14; and/or by (2) modifying the report of the length of the receive window in those same packets; and/or by (2) generating acknowledgements substantially independently of the acknowledgements generated by the end receiving station. The rate control device 26 is preferably placed at the end system acting as the server, but a rate control device may be placed adjacent any system in a path for data. At the server, it is most effective in controlling flow rate because it typically receives and relays the bulk of the traffic of interest.

Several factors are employed to weight the length of the added acknowledgement delay: the amount of data being acknowledged, the measured and smoothed round trip time, the targeted transmission rate, and the mean of the deviation between the measured and smoothed roundtrip time.

The direct feedback rate control mechanism may be incorporated conveniently into computer-executable code, assuming the end system or the router is programmable. It can be used in an open loop environment or it can be used with a rate monitoring mechanism that provides an indication of flow rate.

The following is a detailed description of a specific embodiment of a direct feedback rate control mechanism expressed in pseudocode. For example, the pseudo-code explains a mechanism of generating acknowledgements, including substituting acknowledgements. This pseudo-code also addresses specifics of the TCP environment. It should be noted that latency and the sliding window flow control can be implemented anywhere in the data path.

```
**********************************************************************
Copyright © Packeteer, Inc. 1996

 *
 * tcpSchedule(packet, transport_connection_info, tcpHdr)
 *
 * This function is called for every TCP segment to schedule its emission.
 * An ACK is processed in one direction and DATA in the other direction.
 * Scheduling, as everything else, is directional.
 * NOTE: a "half connection" is one direction of a TCP connection.
 * NOTE: tcpProcessData() has already been called for this segment.
 * NOTE: Use interarrival time of data packets for a Third Order check on
 * detected speed.
 * NOTE: "Our" refers to the local node and its functions.
 *****************************************************************

void
tcpSchedule(PACKET *packet, TCB *tcb, TCP_HDR *tcpHdr)
{

        if there is TCP data, update the average data rate received
        on this HALF connection.

        if (we detected a revised speed estimate or this is the start
            of a new burst) {

                find the CIR/EIR (policy) for this connection/direction
                and update the target rate...
        }

        If no target rate is set, just schedule for immediate transmission
                and return

        if (there is TCP data and it is an OUTBOUND retransmission) {

                 *
                 * Make sure there is no retransmission due to our (local)
                 * own ACK delay...
                 *
                if (we are holding an ACK for this retransmitted sequence
                    which we haven't yet forwarded) {

                        Dispose of packet buffer without sending...
                        return;
                }
        }
        /*
         * Schedule ACK in time per rate...
         */
        if (this ACK sequence is greater than the last forwarded ACK sequence) {
```

```
                 * Do we need to delay this ACK ?
                 */
                if (data rate on other half connection exceeds its target rate) {

                        if (there is TCP data) {
                                /*
                                 * Need to change the ACK on the data packet
                                 * and forward it immediately. Note
                                 * tcpSeparateData() may return a pointer
                                 * to an 'ack packet'...
                                 */
                                if ((packet = tcpSeparateData(tcb, packet, tcpHdr)) == NULL)
                                        return;
                        }
                        /*
                         * Call the downshift function. Note the downshift
                         * function handles ONLY TCP ACK packets (NO TCP data)
                         */
                        tcpDownShift(tcb, packet, tcpHdr, FALSE);
                        return;
                }
                /*
                 * OK to send. Just fall through for immediate scheduling...
                 */
                else {
                        /*
                         * No downshift required...
                         */
                        set lastAckForwarded  on this half connection to tcpHdr->ack;
                        set timeLastFreshAckForwarded  on this half connection to now
                        fall through to transmit immediately

                }
        }
         *
         * Emit immediately...
         *
        Schedule packet for immediate transmission...
}
*******************************************************************

 * tcpSeparateData(tcb, packet, tcpHdr)
 *
 * This is a subroutine called by tcpSchedule().
 * Sometimes we need to pass data with an older ACK.
 * Separate data, assign it an older ack and schedule.
 * Create a new ACK-only packet with specified
 * ACK seq and return it. Update tcpHdr accordingly.
 *
 * This function is not in the performance path.
*******************************************************************
```

```
PACKET *
tcpSeparateData(TCB *tcb, PACKET *packet, TCP_HDR *tcpHdr)
{

        if (there is already an ACK-packet for this connection) {

                copy front of packet (ip & tcp header) into ackPacket

                update the length in the ackPacket (to exclude data)
        }
        else {
                set flag on this half connection to reuse the ackPacket
                when it is eventually recycled, i.e. when transmit is
                completed
        }

        change ACK (and checksum) in original packet and ship it...

        if (we created an ackPacket)
                calculate and set checksum for the ackPacket

        return(ackPacket - may be NULL if we didn't make one...)
}

****************************************************************************

 * tcpDownShift(tcb, packet, tcpHdr, updateChecksum)
 *
 * This subroutine which is called from scheduler here or from the
 * recycle routine.
 * This subroutine scheduled an ACK for a half connection which
 * is exceeding its target bit rate (based on information supplied to it).
 * Push recycle routine to update connection state after ACK is actually emitted.

****************************************************************************

void
tcpDownShift(TCB *tcb, PACKET *packet, TCP_HDR *tcpHdr, BOOL updateChecksum)
{

        If an ack  PACKET is already outstanding for this connection,
        just mark it for reuse   and recycle this packet, and return...

         *
         *  -This is our new ACK packet...
         *
        set this half connections ack packet to point to this packet
        set the appropriate 'recycle' function and argument so we can
        capture the transmit complete confirmation from the driver...
```

```
Check that we are not ACKing more than two MSS worth of data...
originalAmount = amount =  newAck - lastAckForwarded on this connection

if (amount > 2*mss on this half connection) {
        amount = 2*mss;
        newAck = lastAckForwarded  on this connection + amount;
        updateChecksum = TRUE;
}

tsince        = TIME_SINCE(timeLastFreshAckForwarded on this connection);
targetRate    = other half connection target rate
bps           = other half connection smoothed actual data rate
 *
 * Can we downshift all the way in one step...?
 *
ttill =  how much time till we need to send ack for amount bytes at the target rate
 *
 * Clamp TTILL if it appears it will cause an RTO...
 *
if ((trto = tcpRtoTime(tcb, packet, amount)) < ttill) {
         *
         * We can only shift part way - figure partial rate...
         *
        other half connections actual gear rate is set to rate of
                (amount * BITS_PER_BYTE * MSECS_PER_SEC)/trto;
        ttill = trto;
}
else
        other half connection's actual gear rate is set to
        target rate;


other half connection's timeLastShifted =  now

 *
 * If we are late in sending this ACK, see if we can add back any
 * time by ACKing more data...
 *
if (ttill <= tsince) {
         *
         * We are late...
         *
        if (originalAmount > amount) {
                amount = originalAmount;
                reset ttill and new Ack based on new amount
        }
        set lateCarryOver for this half connectionto  reflect how
        late this ack is

         *
         * Make sure we are not late due to window being too small
```

```
		*
		tcpCheckMinWinSize();
	}
	else if there is an existing lateCarryOver for this half connection {
		apply that lateCarryOver to reduce delay here, i.e.
		subtract lateCarryOver from ttill
		set latCarryOver to 0

	}

	delay = (ttill > tsince) ? (ttill-tsince) : 0;

	 *
	 * Do we need to scale down the advertised receive window...
	 *
	if (tcpUpdateAckRxWindow(tcb, packet, originalAmount, delay))
		updateChecksum = TRUE;
	 *
	 * OK - update ACK in packet and checksum if necessary...
	 *
	if (updateChecksum) {
		put  newAck in raw (net format) TCP header...
		and recalculate the TCP checksum
	}

	schedule the ACK packet for transmission at time (now + delay)
}

****************************************************************************
 *
 * tcpSchedAckEmitted(packet)
 *
 * Recycle routine for 'downshifted' ACKs. Update actual ACK in conn info...
****************************************************************************

void
tcpSchedAckEmitted(PACKET *packet)
{
	Make sure connection for this PACKET is still valid
	and still same instance...

	reset the 'ackPacket' for this half connection...

	Update last ack emitted in this direction...

	set lastAckForwarded for this half connection
	set timeLastFreshAckForwarded for this half connection to now

	if (the lastAckForwarded equals the lastAckRcvd for this half connection)
		we are up to date - return
```

```
        Yes - we are reusing it. Compute the next ACK time and reschedule
        this packet.

        tcpHdr.ack     = lastAckRcvd on this half connection
        tcpHdr.dataLen = 0;

        tcpDownShift(tcb, packet, &tcpHdr, TRUE);
}

**********************************************************************

 * tcpRtoTime(tcb, packet, amount)

 * Subroutine called by tcpDownShift()
**********************************************************************

int
tcpRtoTime(TCB *tcb, PACKET *packet, int amount)
{
        return the amount of time until an RTO will occur minus the
        amount of time need to send an ACK to the source of the RTO
        minus a small 'safety' margin. The time till RTO is derived
        from maintaining the smoothed round trip time and mean deviation
        and is essentially 2*mean deviation plus smoothed round trip
        time. This is intended to be a conservative value, many
       implementations may actually set RTO to a greater value.
}

**********************************************************************

 * tcpUpdateAckRxWindow(tcb, packet, amount, delay)
 *
 * Subroutine called by tcpDownshift()
 *
 * Determine whether or not we need to adjust the advertised receive window
 * downward. Returns TRUE if window has been updated.
**********************************************************************

int
tcpUpdateAckRxWindow(TCB *tcb, PACKET *packet, int amount, int delay)
{

        if (the original amount being ACK'ed is greater than 2*MSS AND
            there is no lateCarryOver) {

                if (win closing flag not set) {
                        set win closing flag and return flags, i.e.
                        make adjustment at next ack...
                }
                else {
```

```
                    Set the window size to allow the specified rate
                    given the current RTT and the detected speed.

                    rxWindow = segments *
                      other half connection's target.bps/
                      other half connection's smoothed actual bps;

                *
                * Convert rxwindow from segments to bytes...
                *
                    rxWindow *= mss;

                    reset win closing flag
            }
        }
        else {
                *
                * We are ACKing less than 2*mss
                *
            reset win closing flag;
            return(FALSE);
        }

        clamp rxWindow up to max(rxWindow, 2 * mss)

        check to make sure that the REVISED window size is not
        larger than the actual advertised window

        put the revised window into the packet
        return(TRUE);
}

************************************************************************

 * tcpCheckMinWinSize(si)
 *
 * If we have been LATE with an ack, check to see if we need to pop up our
 * minimum window size...
************************************************************************

#define MIN_WIN_INCREASE_TIME           800
#define MAX_WIN_ADJUST_EFFECT_TIME      1200
#define MAX_WIN_ADJUST_SEGS             4

void
tcpCheckMinWinSize(SEQ_INFO *si)
{
        *
        * Clamp up the minimum window size if we are late here and if
        * we have recently dropped the window size...
        *
```

```
if we have recently reduced the window size and have not recently
   clamped up the minimum window size
        *
        * The longer since we last increased it, the smaller
        * the increase...
        *
        t = min(time since last increase, MIN_WIN_INCREASE_TIME*MAX_WIN_ADJUST_SEGS);
        t = t/MIN_WIN_INCREASE_TIME;
        t = (MAX_WIN_ADJUST_SEGS - t) + 1;
        increase minimum segments by t and set last increase time
        to now
   }
}
*********************************************************************
```

Direct feedback rate control effectively acts as a mechanism to feed back quality of service information to a TCP end system transmitter 12. Direct feedback rate control does not need to store and forward 'ACK only' packets, as these packets can be mechanically created, or even re-used.

In general, scheduling an exact target time for emission for an ACK only packet is non-problematic because the ACK packet is relatively small and is emitted into the flow opposite the prevailing data flow, which is less likely to be instantaneously congested.

Direct feedback rate control is effective for data flowing in either direction, and can even be applied independently to data flowing in either direction on the same connection. When forwarding TCP packets with data, the specific ACK sequence may be changed to control the transmission rate of the reciprocal direction of the connection.

Direct feedback rate control does not buffer or delay and TCP data, hence there is no system requirement for buffering. This helps make direct feedback rate control scaleable to large numbers of simultaneously managed connections.

Direct feedback rate control may increase or decrease the allotted "bandwidth" to any connection on the fly. When doing so, direct feedback rate control ensures that any decrease in transmission rate will be done smoothly and will not incur a retransmission timeout at the transmitter.

Direct feedback rate control will also suppress retransmissions when it is known that their is a 'delayed ack' pending for the retransmitted data.

Direct feedback rate control may pass through multiple equal ACK sequences to allow the 'quick recovery' mechanism to work.

Direct feedback rate control allows for precise, explicit, bi-directional control of individual flows in terms of *committed* and *excess information rate*. Further, committed and excess information rate assignments may be set to scale to a given flow's potential speed, e.g. a T1 user in a given traffic class may be assigned a committed rate of 50 Kbps, where a dial-up user may receive a committed rate of 10 Kbps.

Direct feedback rate control does not build up deep transmission queues or toss packets. It delivers data smoothly and consistently. If a given flow is using bandwidth from the excess pool, that share of excess bandwidth may be reduced on the fly. Direct feedback rate control smoothly downshifts and upshifts flow rates in a manner which does not incur retransmission or delay in end systems.

Direct feedback rate control has additional beneficial characteristics. It smoothens peaks in demand bursts on short time scales. Because of the minimal transmission queue, it allows for delay bounded traffic and does not globally flatten the speed of all connections, as would occur with a deep transmission queue. It allows for the explicit control of individual flows whether or not there is congestion. And it provides a capability to secure network bandwidth from malicious or ill behaved users, e.g. to 'debounce' a remote browser's reload button hits.

Because direct feedback rate control controls inbound as well as outbound bandwidth, administrators can select simple policies such as 'net browsing gets only excess bandwidth at low priority'.

Direct feedback rate control can be used for the following:

As a mechanism to enforce policies for bandwidth allocation of committed and excess information rate assignments.

As a mechanism to manage 'inbound' as well as outbound network bandwidth over a given access link.

As a mechanism to assign explicit throughput rates to individual TCP connections.

As a unique mechanism for feeding back quality of service information to TCP end systems.

It should be noted that an inherent attribute of this invention is the ability to assign rates dynamically, although the control mechanisms for assigning rates are beyond the scope of this disclosure.

The invention has now been explained with reference to specific embodiments. Other embodiments will be apparent to one of ordinary skill in the art. It is therefore not intended that this invention be limited, except as indicated by the appended claims.

WHAT IS CLAIMED IS:

1. A method for controlling data rate of data packets in a digital data packet communication environment having a plurality of digital packet transmission stations interconnectable in a data path, wherein said data packet communication environment lacks explicit end-to-end data rate control, said method comprising the steps of:

causing a first digital packet transmission station at a first end to send a first source packet to a second digital packet transmission station at a second end;

waiting at said first digital packet transmission station for an acknowledgement packet before sending a second source packet;

causing said second digital packet transmission station to send said acknowledgement packet toward said first digital packet transmission station; thereafter

providing a latency in transmission of said acknowledgement packet in the data path between said second digital packet transmission station and said first digital packet transmission station; and thereafter

transmitting from said point of latency said acknowledgement packet to said first digital packet transmission station in order to limit data rate.

2. The method according to claim 1 wherein said digital data packet communication environment employs a flow control window which defines a maximum amount of unacknowledged data packets which a source packet transmission station can permit, further including the step of:

modifying, at said point of latency, the size of said flow control window, said flow control window size being proportional to said desired latency and being greater than is required for receipt of said first source packet.

3. The method according to claim 1 further including:

generating substitute acknowledgement packets at said point of latency in response to groupings of packets from said

first digital packet transmission station, said groupings being generated substantially independently from said acknowledgement packets from said second digital packet transmission station.

4. The method according to claim 1 wherein said data packet communication environment includes end-to-end data flow control.

5. The method according to claim 1 wherein said data packet communication environment employs the Transmission Control Protocol (TCP).

6. The method according to claim 1 wherein said latency providing step is implemented at a third digital packet transmission station between said first digital packet transmission station and said second digital packet transmission station.

7. The method according to claim 1 wherein said latency providing step is implemented at said first digital packet transmission station.

8. The method according to claim 1 wherein said latency providing step is implemented at said second digital packet transmission station.

9. A method for controlling data rate of data packets in a digital data packet communication environment having a plurality of digital packet transmission stations interconnectable in a data path, wherein said data packet communication environment lacks explicit end-to-end data rate control and wherein said digital data packet communication environment employs a flow control window which defines a maximum amount of unacknowledged data packets which a source packet transmission station can permit, said method comprising the steps of:

causing a first digital packet transmission station at a first end to send a first source packet to a second digital packet transmission station at a second end;

waiting at said first digital packet transmission station for an acknowledgement packet before sending a second source packet;

causing said second digital packet transmission station to send said acknowledgement packet toward said first digital packet transmission station; thereafter

providing a latency in transmission of said acknowledgement packet in the data path between said second digital packet transmission station and said first digital packet transmission station;

generating substitute acknowledgement packets at said point of latency in response to groupings of packets from said first digital packet transmission station, said groupings being generated substantially independently from said acknowledgement packets from said second digital packet transmission station;

modifying, at said point of latency, the size of said flow control window, said flow control window size being proportional to said desired latency and being greater than is required for receipt of said first source packet; and thereafter

transmitting from said point of latency said acknowledgement packet to said first digital packet transmission station in order to limit data rate.

10. The method according to claim 9 wherein said data packet communication environment includes end-to-end data flow control.

11. The method according to claim 9 wherein said data packet communication environment employs the Transmission Control Protocol (TCP).

12. The method according to claim 9 wherein said latency providing step is implemented at a third digital packet transmission station between said first digital packet

transmission station and said second digital packet transmission station.

13. The method according to claim 9 wherein said latency providing step is implemented at said first digital packet transmission station.

14. The method according to claim 9 wherein said latency providing step is implemented at said second digital packet transmission station.

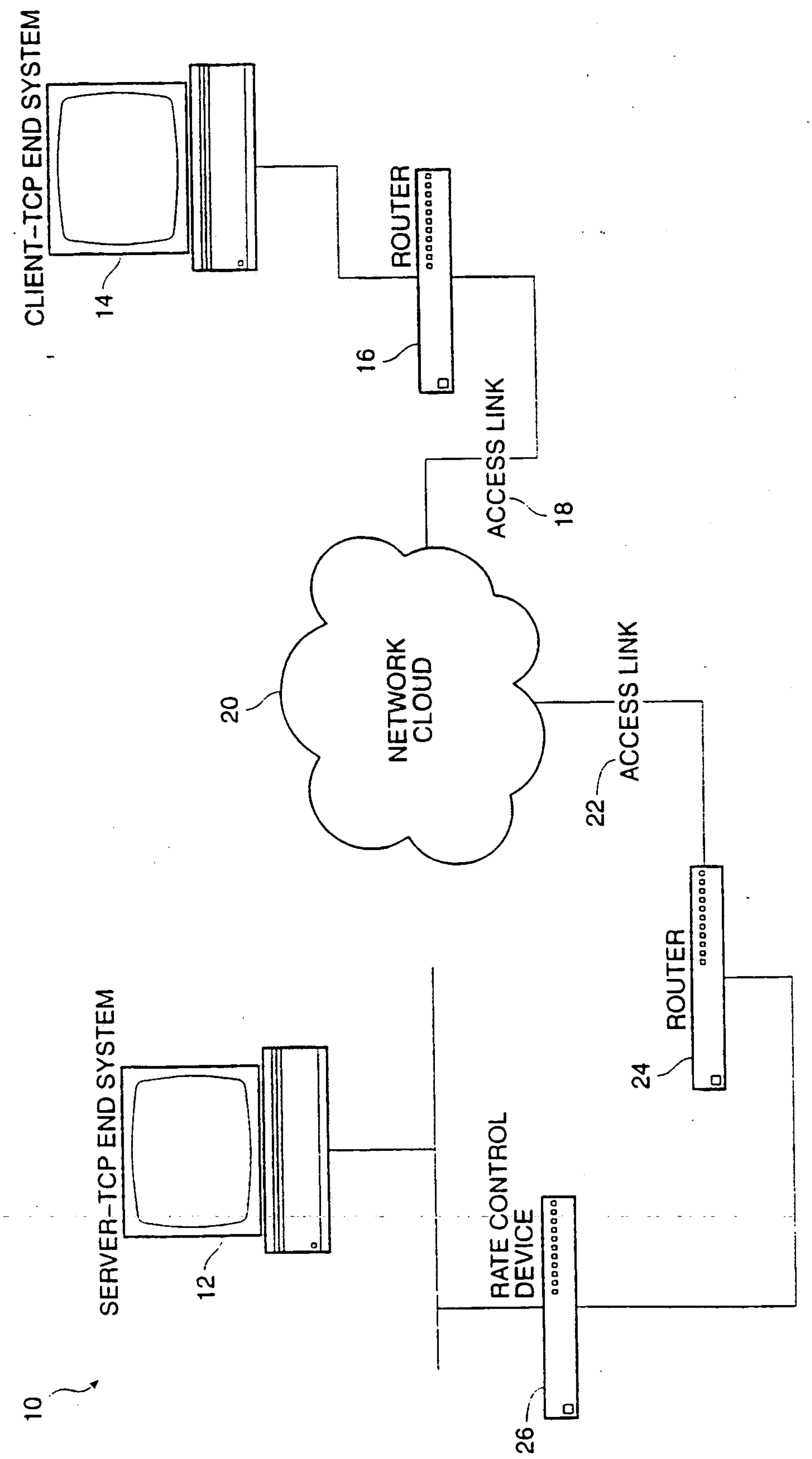

CLIENT-TCP END SYSTEM
14
ROUTER
16
ACCESS LINK
18
NETWORK CLOUD
20
ACCESS LINK
22
ROUTER
24
SERVER-TCP END SYSTEM
12
RATE CONTROL DEVICE
26
10

FIG. 1

**INTERNATIONAL SEARCH REPORT**

International application No.
PCT/US97/19630

**A. CLASSIFICATION OF SUBJECT MATTER**

IPC(6) :H04J 3/26
US CL :370/229, 230, 231, 232, 233, 234, 235, 236, 237, 238
According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 370/229, 230, 231, 232, 233, 234, 235, 236, 237, 238

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, IEEE ONDISC
search terms: acknowledge, window, delay, latency, congestion

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X<br>---<br>Y | US 5,042,029 A (HAYAKAWA) 20 AUGUST 1991, Figures 1 and 5, and column 1, line 51 to column 2, line 6; column 3, lines 14-24; column 4, lines 31-47; and column 6, lines 16-51. | 1, 4, 6-8<br>------------<br>2, 3, 5, 9-14 |
| Y | COMER, D., "Internetworking With TCP/IP", Volume 1, 1991, page 182. | 2, 5, 9-14 |
| Y | THOMAS, S.A., "IPng and the TCP/IP Protocols", 1996, pages 239-240. | 3, 9-14 |

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 06 FEBRUARY 1998 | 03 APR 1998 |

| Name and mailing address of the ISA/US | Authorized officer |
|---|---|
| Commissioner of Patents and Trademarks<br>Box PCT<br>Washington, D.C. 20231 | MELVIN MARCELO |
| Facsimile No. (703) 305-3230 | Telephone No. (703) 305-3900 |

Form PCT/ISA/210 (second sheet)(July 1992)★

CORRECTED VERSION*

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

**(51) International Patent Classification 6 :** H04J 3/26

A1

**(11) International Publication Number:** WO 98/20511

**(43) International Publication Date:** 14 May 1998 (14.05.98)

**(21) International Application Number:** PCT/US97/19630

**(22) International Filing Date:** 24 October 1997 (24.10.97)

**(30) Priority Data:**
08/742,994 1 November 1996 (01.11.96) US

**(71) Applicant:** PACKETEER, INC. [US/US]; Suite 305, 307 Orchard City Drive, Campbell, CA 95008 (US).

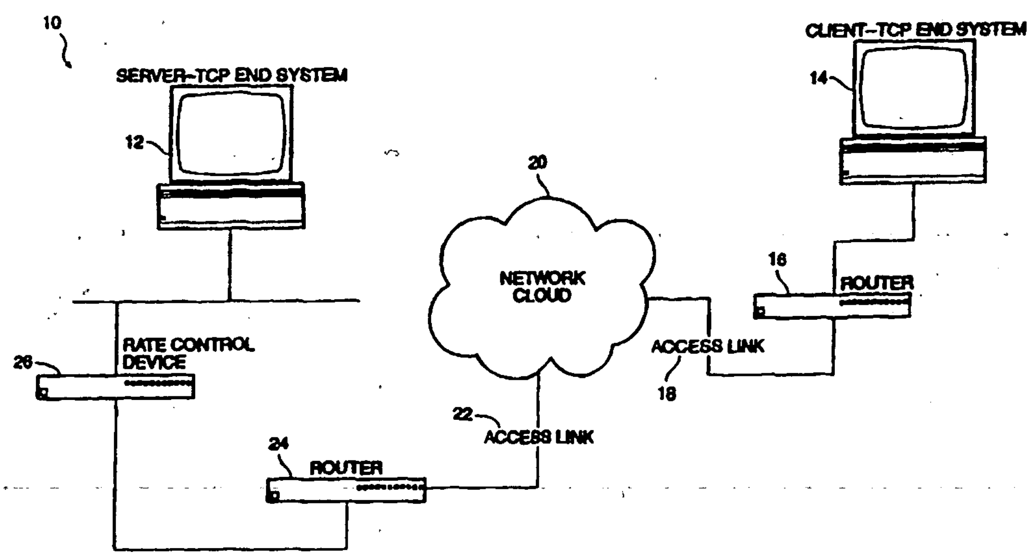
**(72) Inventor:** PACKER, Robert, L.; 16095 Redwood Lodge Road, Los Gatos, CA 95036 (US).

**(74) Agents:** ALLEN, Kenneth, R. et al.; Townsend and Townsend and Crew LLP, 8th floor, Two Embarcadero Center, San Francisco, CA 94111-3834 (US).

**(81) Designated States:** AU, CN, IL, JP, KR, NZ, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

**Published**
*With international search report.*
*Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.*

**(54) Title:** METHOD FOR EXPLICIT DATA RATE CONTROL IN A PACKET COMMUNICATION ENVIRONMENT WITHOUT DATA RATE SUPERVISION




**(57) Abstract**

A method for explicit data rate control is introduced into a packet communication environment (10) which does not have data rate supervision by adding latency to the acknowledgement (ACK) packet and by adjusting the size of the flow control window associated with the packet in order to directly control the data rate of the source data at the station (12 or 14) originating the packet.

*FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **AL** | Albania | **ES** | Spain | **LS** | Lesotho | **SI** | Slovenia |
| **AM** | Armenia | **FI** | Finland | **LT** | Lithuania | **SK** | Slovakia |
| **AT** | Austria | **FR** | France | **LU** | Luxembourg | **SN** | Senegal |
| **AU** | Australia | **GA** | Gabon | **LV** | Latvia | **SZ** | Swaziland |
| **AZ** | Azerbaijan | **GB** | United Kingdom | **MC** | Monaco | **TD** | Chad |
| **BA** | Bosnia and Herzegovina | **GE** | Georgia | **MD** | Republic of Moldova | **TG** | Togo |
| **BB** | Barbados | **GH** | Ghana | **MG** | Madagascar | **TJ** | Tajikistan |
| **BE** | Belgium | **GN** | Guinea | **MK** | The former Yugoslav Republic of Macedonia | **TM** | Turkmenistan |
| **BF** | Burkina Faso | **GR** | Greece | | | **TR** | Turkey |
| **BG** | Bulgaria | **HU** | Hungary | **ML** | Mali | **TT** | Trinidad and Tobago |
| **BJ** | Benin | **IE** | Ireland | **MN** | Mongolia | **UA** | Ukraine |
| **BR** | Brazil | **IL** | Israel | **MR** | Mauritania | **UG** | Uganda |
| **BY** | Belarus | **IS** | Iceland | **MW** | Malawi | **US** | United States of America |
| **CA** | Canada | **IT** | Italy | **MX** | Mexico | **UZ** | Uzbekistan |
| **CF** | Central African Republic | **JP** | Japan | **NE** | Niger | **VN** | Viet Nam |
| **CG** | Congo | **KE** | Kenya | **NL** | Netherlands | **YU** | Yugoslavia |
| **CH** | Switzerland | **KG** | Kyrgyzstan | **NO** | Norway | **ZW** | Zimbabwe |
| **CI** | Côte d'Ivoire | **KP** | Democratic People's Republic of Korea | **NZ** | New Zealand | | |
| **CM** | Cameroon | | | **PL** | Poland | | |
| **CN** | China | **KR** | Republic of Korea | **PT** | Portugal | | |
| **CU** | Cuba | **KZ** | Kazakstan | **RO** | Romania | | |
| **CZ** | Czech Republic | **LC** | Saint Lucia | **RU** | Russian Federation | | |
| **DE** | Germany | **LI** | Liechtenstein | **SD** | Sudan | | |
| **DK** | Denmark | **LK** | Sri Lanka | **SE** | Sweden | | |
| **EE** | Estonia | **LR** | Liberia | **SG** | Singapore | | |